Heuristic Search: Let $e(v)$ be an estimate of the distance from $v$ to the goal $t$.

Use Dijkstra's algorithm with $d(v) + e(v)$ as the selection criterion.

The method works if

$$e(v) \leq L(v, w) + e(w) \text{ for all } v, w$$

(Estimate $e$ is a consistent lower bound on the actual distance.)

In Euclidean graphs the distance “as the crow flies” works.

Hart, Nilsson, Rafael (1968)
Dijkstra's algorithm

Heuristic search
Bidirectional Search: Search forward from $s$ and backward from $t$ concurrently.

$\Rightarrow$ Getting the stopping rule correct is tricky, especially for bidirectional heuristic search.
The Minimum Spanning Tree Problem

Given a connected graph, find a spanning tree of minimum total edge cost.

where,

\[ n = \text{the number of vertices} \]

\[ m = \text{the number of edges} \]

\[ n - 1 \leq m \leq \binom{n}{2} \]
Applications

Network Construction

Clustering

Minimum Tour Relaxation (Held-Karp 1-trees)
A Simple Solution From the 80's
(with apologies to Oliver Stone)

Gorden Gecko: "Greed is Good"

Repeatedly select the cheapest unselected edge and add it to the tree under construction if it connects two previously disconnected pieces.

Kruskal, 1956
The greedy method generalizes to matroids.

We shall generalize the method rather than the domain of application.
Generalized Greedy Method

Beginning with all edges uncolored,
sequentially color edges
blue (accepted) or red (rejected).

Blue Rule:
Color blue any minimum-cost uncolored edge crossing a cut with no blue edges crossing.

Red Rule:
Color red any maximum-cost uncolored edge on a cycle with no red edges.
"Classical" Algorithms

(before algorithm analysis)

Kruskal’s algorithm, 1956

$O(m \log n)$ time

Jarnik’s algorithm, 1930

$O(n^2)$ time

also Prim, Dijkstra

Boruvka’s algorithm, 1926

$O(\min\{m \log n, n^2\})$ time

and many others
Jarnik’s Algorithm

Grow a tree from a single start vertex.
At each step add a cheapest edge with exactly one end in the tree.
Boruvka's Algorithm

Repeat the following step until all vertices are connected:

For each blue component, select a cheapest edge connecting to another component; color all selected edges blue.

For correctness, a tie-breaking rule is needed.

Henceforth, assume all edge costs are distinct.

Then there is a unique spanning tree.
Selected History

Boruvka, 1926

Jarnik, 1930
Prim, 1957
Dijkstra, 1959

Kruskal, 1956

Williams, Floyd, 1964
heaps

Yao, 1975
packets in Boruvka’s algorithm

Fredman, Tarjan, 1984
F-heaps in:
Jarnik’s algorithm
a hybrid Jarnik-Boruvka algorithm

Gabow, Galil, Spencer, 1984
Packets in F-T algorithm

\[ \log^* n = \min \{ i \mid \log \log \log \ldots \log n \leq 1 \} \]

where the logarithm is iterated \( i \) times
Models of Computation

We assume comparison of the two edge costs takes unit time, and no other manipulation of edge costs is allowed.

Another model:

bit manipulation of the binary representations of edge costs is allowed.

In this model,

Fredman-Willard, 1990, achieved $O(m)$ time.

(fast small heaps by bit manipulation)
Goal: An $O(m)$-time algorithm
without bit manipulation of edge weights

Boruvka’s algorithm with contraction:

If $G$ contains at least two vertices:

select cheapest edge incident to each vertex;

Contract all selected edges;

Recur on contracted graph.

If contraction preserves sparsity ($m = O(n)$),

this algorithm runs in $O(n) = O(m)$ time

on sparse graphs.

E.g. planar graphs
How to handle non-sparse graphs?

Thinning: remove all but $O(n)$ edges by finding edges that can’t be in the minimum spanning tree.

How to thin?